UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Disciplina	Turmas
Algoritmos e Estruturas de Dados II	

Professores

Data de entrega: 19/05/2016 até as 23:55

Execício de Programação em Linguagem C (TP1)

O jovem Teseu, um bravo guerreiro ateniense, decidiu enfrentar o Minotauro da Ilha de Creta. Para que ele tenha alguma chance de derrotar o Minotouro, ele necessita encontrar uma espada mágica dentro do Labirinto de forma rápida e sem se perder. Para que consiga completar essa difícil e perigosa tarefa com sucesso ele precisa de sua ajuda. Teseu, sabendo de suas incríveis habilidades como programador, precisa que você escreva um programa que, dado o mapa do labirinto e a localização da espada dentro do mesmo, retorne o caminho até ela, se possível.

O problema

O labirinto é dado por uma matriz binária $N \times N$ onde a entrada e a localização da espada são dados pelas variáveis E e S respectivamente, onde E e S são pares ordenados do formato (x,y). Teseu inicia sua caminhada da entrada E até local S onde a espada se encontra, e com um fio de linha ele vai marcando o caminho para não se perder. Teseu pode mover-se apenas em quatro direções: direita, esquerda, cima e baixo (Note que não há movimento em diagonal). No mapa do labirinto, **1 significa que há uma parede naquela coordenada e 0 significa que há passagem**. Dado esse mapa, seu algoritmo deve retornar uma outra matriz binária com as mesmas dimensões do labirinto onde os valores 1 representam o caminho encontrado até a espada.

Entrada

A entrada do problema deve ser lida de um arquivo texto com o seguinte formato:

```
N x y sx sy
L(0,0) L(0,2) L(0,3) ... L(0,N-1)
L(1,0) L(1,2) L(0,3) ... L(1,N-1)
                ...
L(N-1,0) L(N-1,2) L(N-1,3) ... L(N-1, N-1)
```

Onde N é dimensão do labirinto, x e y são as coordenadas da entrada do labirinto, sx e sy são as coordenadas da espada. As entradas L(i,j) indicam o valor da casa na linha i e coluna j do labirinto.

Segue um exemplo de arquivo de entrada com dados reais:

```
6 1 0 4 4
1 1 1 1 1 1
0 0 1 1 1 1
1 0 0 1 0 1
1 1 0 1 1 1
1 0 0 0 0 1
1 1 1 1 1 1
```

Note que nesse problema o mapa do labirinto é modelado como um tabuleiro para uma maior abstração do problema. A figura 1 mostra mais claramente como seria essa abstração. As casas cinzas (valor 1) são as paredes e as brancas (valor 0) são as passagens.

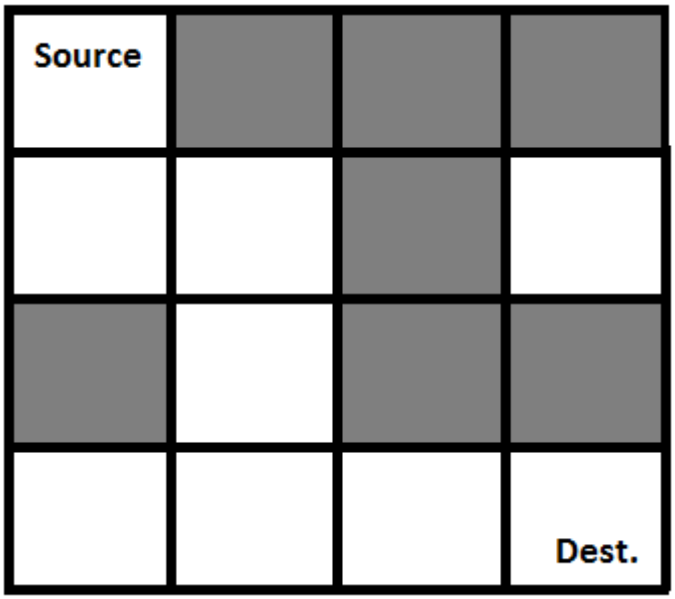

Figura 1: Exemplo de um labirinto

Saída

Seu programa deve escrever a solução em um arquivo do seguinte modo:

```
0 0 0 0 0 0
1 1 0 0 0 0
0 1 1 0 0 0
0 0 1 0 0 0
0 0 1 1 1 0
0 0 0 0 0 0
```

Os elementos com valor 1 representam onde Teseu deve passar para que ele chegue até a espada. A figura 2 ilustra a solução encontrada. Além disso, *caso não haja um caminho válido até a espada o programa deve gravar no arquivo de saída um único valor 0.*

É importante observar que para algumas entradas existe mais de uma solução possível. Você não precisa se preocupar em escolher algum caminho específico, desde que sua solução apresentada na saída seja correta (leve Teseu até a espada e sem transpassar paredes).

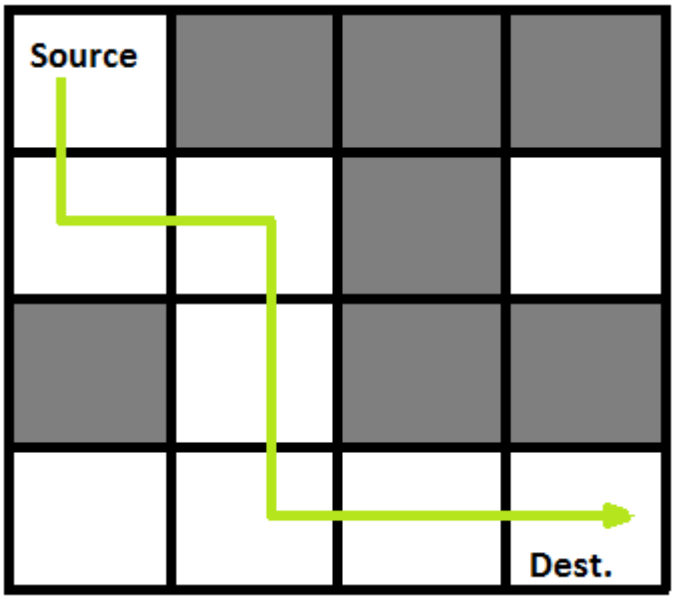

Figura 2: Exemplo de um labirinto com caminho

Algoritmo

Para realizar o algoritmo é necessário o mapa do labirinto, a coordenada (x,y) da entrada e a coordenada (sx, sy) do local da espada. Começando pela entrada (x, y), o algoritmo deverá marcar a casa atual com valor 1 e visitar as casas vizinhas. O algoritmo visita a casa da direita e verifica, recursivamente, se esse vizinho leva a uma solução válida. Caso não leve, ele verifica o próximo vizinho até que reste mais nunhum. Se nenhuma das soluções anteriores levarem a um caminho válido o algoritmo retorna falso.

Abaixo o pseudo-código do algoritmo recursivo para encontrar a espada no labirinto. Você poderá basear sua implementação neste exemplo.

Algorithm 1 Caminha no labirinto recursivamente

```
function CAMINHALABIRINTO(mapa, x, y, sx, sy, sol)
    if x = sx && y = sy then
        sol[x][y] ← 1
        return True
    else
        if mapa[x][y] = 0 && sol[x][y] = 0 then
            sol[x][y] ← 1
            if caminhalabirinto(mapa, x + 1, y, sx, sy, sol) = True then
                return True
            end if
            if caminhalabirinto(mapa, x, y + 1, sx, sy, sol) = True then
                return True
            end if
            if caminhalabirinto(mapa, x − 1, y, sx, sy, sol) = True then
                return True
            end if
            if caminhalabirinto(mapa, x, y − 1, sx, sy, sol) = True then
                return True
            end if
            sol[x][y] ← 0
        end if
    end if
    return True
end function
```

Estrutura de dados

A estrutura de dados a seguir deve ser utilizada para armazenar o mapa do labirinto após a leitura do arquivo.

```
typedef struct {
   int N;         // Dimensão do labirinto, lmebre-se que o mesmo é N x N
   int x;         // Coordenada x da entrada
   int y;         // Coordenada y da entrada
   int sx;        // Coordenada x da espada
   int sy;        // Coordenada y da espada
   unsigned char **mapa;     // variável para armazenar o mapa (matriz)
} Labirinto;
```

O que deve ser feito

1. Elaborar um programa capaz de ler um arquivo texto com a representação do labirinto e imprimir no arquivo a solução com o trajeto até a espada.

   `./exec labirinto.txt caminho_espada.txt alg`

   onde *exec* é o nome do executável, `labirinto.txt` contém o mapa do labirinto com as posições de Teseu e da espada, `caminho_espada.txt` receberá a saída do algoritmo contendo o caminho encontrado até a espada e `alg` é um inteiro indicando se o algoritmo será recursivo (valor 0) ou, então, iterativo (valor 1).

2. O programa deve ter a função `Labirinto* LeLabirinto(const char * entrada)` que lê um arquivo no formato descrito anteriormente e armazena seus dados em uma variável do tipo Labirinto.

3. O programa deve ter a função `int CaminhaLabirintoRecursivo(Labirinto* lab, int x, int y, int ** sol)` que computa o caminho entre a entrada do labirinto até o local que a espada se encontra. O caminho encontrado é amarzenado na matriz *sol* e a função deve retornar o valor *1* caso haja um caminho válido e *0* caso contrário. Note que essa função deve ser implementada de forma recursiva.

4. O programa deve ter a função `int CaminhaLabirintoIterativo(Labirinto* lab, int x, int y, int ** sol)` que computa o caminho entre a entrada do labirinto até o local que a espada se encontra. O caminho encontrado é amarzenado na matriz *sol* e a função deve retornar o valor *1* caso haja um caminho válido e *0* caso contrário. Essa função deve ser implementada de forma iterativa utilizando a estrutura Pilha implementada por você para simular a recursão do algoritmo ao visitar cada casa do labirinto e suas vizinhas. Ou seja, a ordem de visitação deve ser a mesma do algoritmo recursivo e, portanto, produzir a mesma solução.

5. OBS: Utilizar um vetor de tamanho fixo para armazenar os dados da pilha causa desperdício de memória quando poucos dados são colocados na pilha. *Logo, implemente a pilha usando alocação encadeada.*

6. Todas as funções implementadas devem possuir um cabeçalho conforme o exemplo a seguir:

```
/*-----------------------------------------------------------------
Prototipo: CaminhaLabirintoRecursivo(Labirinto* lab, int x,...
Funcao: Computa o caminho entre a entrada ...
Entrada: Estrutura contendo....
Saida: ...
-----------------------------------------------------------------*/
```

7. Seu programa não deverá ter nenhum *leak* de memória, ou seja, tudo que for alocado de forma dinâmica, deverá ser desalocado com a função "free()"antes do término da execução.

8. O programa deverá ser possível de ser compilado no Linux, portanto ele não deverá conter nenhuma biblioteca que seja específica do sistema operacional Windows.

9. Você deverá implementar o trabalho na IDE Code::Blocks (*disponível em: http://www.codeblocks.org/*). Deve ser submetido ao moodle o diretório do projeto criado no Code::Blocks em um arquivo compactado contendo os arquivos ".h"(com as declarações das funções) e ".c"(um com as implementações das funções descritas nesse documento e outro com a função *main*. Além disso, a submissão também deverá conter o relatório do trabalho em formato PDF.

10. O trabalho deverá ser entregue via *moodle* até as 23:55 horas do dia 19/05/2016. **Trabalhos que forem entregues fora do prazo não serão aceitos em nenhuma hipótese.**

Documentação

Escreva um documento explicando o seu código e avaliando o desempenho de sua implementação. Separe-o em cinco seções: introdução, implementação, resultados, conclusão e referências. Seja claro e objetivo.

- **Introdução**: Faça uma breve introdução o problema, definindo-o com as *suas* palavras.
- **Implementação**: Explique quais foram as estratégias adotadas para a leitura do arquivo de entrada, realização do algoritmo de busca recursivo e iterativo e a escrita da solução em disco. Descreva qual o papel de cada função, seus parâmetros de entrada e de saída.
- **Resultados**: Faça testes com seu programa utilizando diversas labirintos, posições da entrada e espada. Apresente o resultado obtido e faça uma breve discussão sobre eles. (Serão disponibilizadas dois labirintos para teste). **Faça a análise da sua implementação em termos de tempo e espaço indicando as funções e as ordens de complexidade de cada um. Responda as perguntas: o seu algoritmo tem complexidade polinomial ou exponencial? Por que? Você consegue pensar em alguma forma de resolver o problema do labirinto com um custo mais baixo do que o algoritmo passado neste documento?**
- **Conclusão**: Explique quais foram as dificuldades encontradas durante o desenvolvimento e as conclusões obtidas.
- **Referências**: Se você utilizou informações adicionais além das especificadas neste trabalho, cite as fontes.

Avaliação

- **4 pontos** pela implementação, onde serão avaliados, além do funcionamento adequado do programa: identação correta do código, comentários das funções, alocações de desalocações dinâmicas bem feitas e modularização.
- **4 pontos** pelos testes, onde serão avaliados os resultados obtidos.
- **6 pontos** pela documentação, onde serão avaliados a clareza das seções e a discussão dos resultados obtidos.
- ***Lembramos que será utilizado um software de detecção de cópias e qualquer tipo de plágio detectado resultará em nota 0 para ambos trabalhos!***